

```
1  #include "Game.h"
2  #include <conio.h>
3  #include <Windows.h>
4  #include <iostream>
5
6  #include "Enemy.h"
7  #include "Key.h"
8  #include "Door.h"
9  #include "Money.h"
10 #include "Goal.h"
11
12 using namespace std;
13
14 constexpr int kArrowInput = 224;
15 constexpr int kLeftArrow = 75;
16 constexpr int kRightArrow = 77;
17 constexpr int kUpArrow = 72;
18 constexpr int kDownArrow = 80;
19 constexpr int kEscapeKey = 27;
20 constexpr int kBackspace = 8;
21
22 Game::Game()
23     : gameOver{ false } {};
24
25 Game::~~Game() {};
26
27 bool Game::load()
28 {
29     return level1.LoadLevel("Level1.txt", player1.GetXPositionPointer(),
30                             player1.GetYPositionPointer());
31 }
32
33 void Game::Run()
34 {
35     Draw();
36     gameOver = Update();
37
38     if (gameOver)
39     {
40         Draw();
41     }
42 }
43
44 bool Game::isGameOver()
45 {
46     return gameOver;
47 }
48
49 bool Game::Update()
50 {
51     int input = _getch();
52     int arrowInput = 0;
53     int newPlayerX = player1.GetXPosition();
54     int newPlayerY = player1.GetYPosition();
```

```
53
54     // One of the Arrow keys were pressed
55     if (input == kArrowInput)
56     {
57         arrowInput = _getch();
58     }
59
60     if ((input == kArrowInput && arrowInput == kRightArrow) ||
61         ((char)input == 'd' || (char)input == 'D'))
62     {
63         newPlayerX++;
64     }
65
66     if ((input == kArrowInput && arrowInput == kLeftArrow) ||
67         ((char)input == 'a' || (char)input == 'A'))
68     {
69         newPlayerX--;
70     }
71
72     if ((input == kArrowInput && arrowInput == kUpArrow) ||
73         ((char)input == 'w' || (char)input == 'W'))
74     {
75         newPlayerY--;
76     }
77
78     if ((input == kArrowInput && arrowInput == kDownArrow) ||
79         ((char)input == 's' || (char)input == 'S'))
80     {
81         newPlayerY++;
82     }
83
84     if (input == kEscapeKey)
85     {
86         userQuit = true;
87         return true;
88     }
89     if ((char)input == 'Z' || (char)input == 'z')
90     {
91         player1.DropKey();
92     }
93
94     //If position never changed
95
96     if (newPlayerX == player1.GetXPosition() && newPlayerY ==
97         player1.GetYPosition())
98     {
99         return false;
100     }
101     else
102     {
103         return HandleCollision(newPlayerX, newPlayerY);
104     }
104 }
```

```
105
106 bool Game::HandleCollision(int newPlayerX, int newPlayerY)
107 {
108     PlaceableActor* collidedActor = level1.UpdateActors(newPlayerX,           ↗
newPlayerY); // creates a placeable actor
109     if (collidedActor != nullptr && collidedActor->IsActive())
110     {
111         Enemy* collidedEnemy = dynamic_cast<Enemy*>(collidedActor); //       ↗
specifies the type/ thing we are trying to cast, in this case an ↗
enemy
112         if (collidedEnemy)
113         { // if the pointer is valid, if statement works, if it is a key       ↗
none of the code will work
114             collidedEnemy->Remove(); // if a collision with an enemy occurs, ↗
the enemy is removed.
115             player1.SetXYPosition(newPlayerX, newPlayerY); // players       ↗
position is set to new position
116
117             player1.DecrementLives(); // decrmeent lives
118             if (player1.GetLive() < 0) // if less than zero game is over.
119             {
120                 return true; // game is over
121             }
122         }
123         Money* collidedMoney = dynamic_cast<Money*>(collidedActor); // if     ↗
collided with money
124         if (collidedMoney)
125         {
126             collidedMoney->Remove(); // remove the money
127             player1.AddMoney(collidedMoney->GetWorth()); // add the money     ↗
and show the worth.
128             player1.SetXYPosition(newPlayerX, newPlayerY);
129         }
130         Key* collidedKey = dynamic_cast<Key*>(collidedActor);
131         if (collidedKey)
132         {
133             if (!player1.HasKey())
134             {
135                 player1.PickUpKey(collidedKey);
136                 collidedKey->Remove();
137                 player1.SetXYPosition(newPlayerX, newPlayerY);
138             }
139         }
140         Door* collidedDoor = dynamic_cast<Door*>(collidedActor);
141         if (collidedDoor)
142         {
143             if (!collidedDoor->IsOpen())
144             {
145                 if (player1.HasKey(collidedDoor->GetColour()))
146                 {
147                     collidedDoor->Open();
148                     collidedDoor->Remove();
149                     player1.UseKey();
```

```
150         player1.SetXYPosition(newPlayerX, newPlayerY);
151     }
152     else
153     {
154
155     }
156 }
157 else
158 {
159     player1.SetXYPosition(newPlayerX, newPlayerY); // player goes through the door
160 }
161 }
162 Goal* collidedGoal = dynamic_cast<Goal*>(collidedActor);
163 if (collidedGoal)
164 {
165     collidedGoal->Remove(); // removes actors
166     player1.SetXYPosition(newPlayerX, newPlayerY);
167     return true;
168 }
169 }
170 else if (level1.IsSpace(newPlayerX, newPlayerY))
171 {
172     player1.SetXYPosition(newPlayerX, newPlayerY);
173 }
174 else if (level1.IsWall(newPlayerX, newPlayerY))
175 {
176     // wall collision
177 }
178 return false;
179 }
180
181 void Game::Draw()
182 {
183     HANDLE console = GetStdHandle(STD_OUTPUT_HANDLE);
184     system("cls");
185
186     level1.Draw();
187
188     //Set cursor position for player
189     COORD actorCursorPosition;
190     actorCursorPosition.X = player1.GetXPosition();
191     actorCursorPosition.Y = player1.GetYPosition();
192     SetConsoleCursorPosition(console, actorCursorPosition);
193     player1.Draw();
194
195
196     //Set cursor to end of level.
197     COORD currentCursorPosition;
198     actorCursorPosition.X = 0;
199     actorCursorPosition.Y = level1.GetHeight();
200     SetConsoleCursorPosition(console, actorCursorPosition);
201 }
```